Gerenciando o desenvolvimento de uma composição de Serviços Web Semânticos através do OWL-S Composer

André de Amorim Fonseca[1], **Daniela Barreiro Claro**[1], **Denivaldo Cicero Pavão Lopes**[2]

[1]Laboratório de Sistemas Distribuídos - LaSiD
Departamento de Ciência da Computação - Universidade Federal da Bahia
Av. Adhemar de Barros, s/n. Salvador/BA – Brasil CEP. 40170-110

deco.af@gmail.com, dclaro@ufba.br
[2]Departamento de Engenharia de Eletricidade - Universidade Federal do Maranhão
Avenida dos Portugueses s/n. Sao Luis/MA – Brasil CEP.65080-040

denivaldo.lopes@gmail.com

Resumo. *A diversidade de Sistemas de Informação que tem utilizado a Web demanda uma crescente utilização dos serviços Web para facilitar a interoperabilidade destes sistemas. Atualmente, muitos serviços Web simples não atendem mais às necessidades destes sistemas, tendo que ser combinados dando origem às composições de serviços Web (Web service Composition). Ambientes integrados vêm incorporando funcionalidades para facilitar o desenvolvimento de um serviço Web simples, como o plugin WTP(Web Tool Plugin) no Eclipse, porém poucos têm se preocupado com as composições de serviços. Além disso, dos ambientes propostos que tratam as composições semânticas, nenhum deles propõe reutilizar ferramentas já existentes para compor os serviços Web, reduzindo tanto o tempo de desenvolvimento da composição quanto o tempo de aprendizado do usuário. Assim, o presente trabalho tem dois principais objetivos: analisar os requisitos fundamentais para compor serviços Web semânticos; e, de posse destes requisitos, desenvolver um* plugin *para o Eclipse denominado OWL-S Composer para compor serviços Web semânticos através de uma interface gráfica intuitiva e amigável, reutilizando ferramentas e gerando uma composição correta, tanto sintática quanto semanticamente.*

Abstract. *The diversity of Information Systems that use the Web has demand an increasing use of Web services to facilitate the interoperability of these systems. Currently, many Web services simply do not meet the needs of these systems, needing to be combined leading to the Web services Compositions. Integrated environments are incorporating features to facilitate the development of a simple Web service, such as the Eclipse plugin WTP(Web Tool Plugin), but only a few of them have been concerned with the Web Services Composition. Moreover, about the environments that address the semantic compositions, none of them proposes the reuse of existing tools to make the Web services Composition, reducing both the time of the composition development and the learning time of the user. Thus, this work has two main objectives: to analyze the fundamental requirements to make the Web services Composition, and, in the possession of these requirements, develop* plugin *for Eclipse called OWL-S Composer to compose semantic Web services through a intuitive and friendly user interface, reusing tools and creating a proper composition, both syntactic as semantically.*

1. Introdução

O crescente desenvolvimento de Sistemas de Informação requer a utilização de ferramentas amigáveis e funcionais para facilitar e agilizar o processo de construção destes sistemas. A utilização da Web como forma de publicar novas aplicações e disponibilizar novas funcionalidades tem consolidado o uso dos serviços Web. Estes serviços vêm sendo incorporados em muitos Sistemas de Informação e têm garantido assim a interoperabilidade entre sistemas heterogêneos[Claro and Macêdo 2008b]. A grande vantagem desta interoperabilidade é que estes serviços podem ser combinados dando origem a outros serviços Web. Assim, uma composição de serviços Web (*Web Service Composition*) pode ser definida como um novo serviço Web resultante da combinação de outros serviços isolados.

A utilização de ferramentas amigáveis e funcionais no processo de desenvolvimento de um serviço Web já foi incorporado em diversos ambientes integrado de desenvolvimento (*IDE - Integrated Development Environment*) como o Eclipse ([Eclipse 2003]). Estas ferramentas podem ser desenvolvidas em formato de *plugins*, como por exemplo, a utilização do WTP(*Web Tool Plugin*)[Dai et al. 2007] no ambiente Eclipse. O WTP é um *plugin* que transforma o desenvolvimento de um serviço Web, outrora árduo, trabalhoso e demorado, em um processo fácil e rápido, ampliando a utilização deste ambiente para o desenvolvimento de aplicações Web. Na verdade, a utilização de um ambiente de desenvolvimento integrado aumenta a produtividade e diminui a curva de apredizagem de um desenvolvedor Web, contribuindo para o desenvolvimento de sistemas de informação novos e integrados.

Algumas outras ferramentas ou *plugins* já foram desenvolvidos dentro de ambientes integrados com o intuito de agilizar e facilitar a manipulação das composições de serviços [Scicluna et al. 2004, Srinivasan et al. 2006, Chafle et al. 2007, Knublauch et al. 2004]. Porém, muitos destes trabalhos não utilizam componentes da própria ferramenta, não inicializam o processo de desenvolvimento de composições por meios gráficos e nem criam os arquivos de saída da composição de uma maneira sintaticamente e semanticamente correta, exigindo em muitos casos o conhecimento de outras tecnologias por parte do usuário e um tempo extra na compreensão do código gerado. Muitas vezes, estes fatores desencorajam os desenvolvedores a utilizar estes ambientes e assim continuam utilizando o processo manual de criação das composições, retardando o processo de desenvolvimento de sistemas.

Diante deste contexto, este trabalho propõe o desenvolvimento de um *plugin* dentro de um ambiente integrado como o Eclipse para gerenciar o desenvolvimento de uma composição de serviços Web utilizando serviços previamente desenvolvidos e publicados na Web, garantindo um arquivo de saída sintaticamente e semanticamente correto. Além disso, este trabalho adiciona características semânticas aos serviços desenvolvidos, facilitando assim a sua descoberta e, consequentemente, permitindo-o que a sua descrição seja realizada de uma maneira não-ambígua. As principais contribuições deste artigo podem ser resumidas abaixo:

- propor e analisar os principais requisitos para a composição de serviços Web utilizando um ambiente integrado;
- utilizar ferramentas existentes (WTP no Eclipse) para facilitar o desenvolvimento dos serviços a serem incluídos na composição;

- criar uma ferramenta (*plugin*) para o desenvolvimento de composições de serviços semânticos utilizando uma interface gráfica intuitiva e amigável;
- gerar o arquivo OWL-S referente à composição proposta pelo usuário de forma que o mesmo seja sintaticamente e semanticamente correto.

O presente artigo está assim organizado: a seção 2 apresenta as composições de serviços, incluindo a linguagem OWL-S. A seção 3 propõe alguns requisitos fundamentais para o desenvolvimento de um *plugin*. A seção 4 introduz o *plugin* proposto: *OWL-S Composer*. A seção 5 relaciona alguns trabalhos da literatura com o *OWL-S Composer*. A seção 6 apresenta alguns experimentos realizados e resultados obtidos e a seção 7 conclui e direciona alguns trabalhos futuros.

2. Composição de serviços Web

Diversos protótipos desenvolvidos propõem duas etapas para o processo de desenvolvimento de uma composição de serviços Web [Claro and Macêdo 2008a]. Uma primeira etapa conhecida como abstrata ou planejamento da composição que determina as atividades que a composição deve realizar e uma segunda etapa conhecida como concreta ou executora, onde efetivamente a composição pode ser executada. Composições abstratas estão sendo utilizadas juntamente com outras áreas de pesquisa como planejadores [Digiampietri et al. 2007] com o intuito de determinar automaticamente quais atividades devem participar de uma composição de serviço. Embora a versão atual deste trabalho não contemple as composições automáticas, os serviços, neste trabalho, foram projetados para que esta funcionalidade fosse incorporada. Nesse contexto, é importante a utilização de linguagens semânticas de descrição dos serviços Web como OWL-S [Martin et al. 2007] e WSMO[Domingue et al. 2005]. Dentre estas, a OWL-S harmoniza de forma satisfatória a interação entre os serviços Web (coreografia) e a visão global da composição, ou seja, o agrupamento dos serviços dado um objetivo proposto (orquestração), sendo que, a mesma carece de ferramentas amigáveis para popularizar a sua utilização. Assim, este trabalho utiliza a OWL-S como linguagem de descrição dos serviços que participam da composição assim como do serviço composto gerado como resultado. A figura 1 demonstra as entradas e saídas da ferramenta proposta para o gerenciamento da composição.

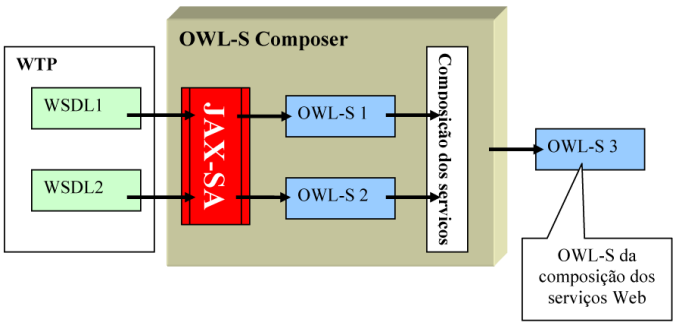

Figura 1. As entradas e saídas do *OWL-S Composer*

O WSDL 1 assim como o WSDL 2 presente na figura 1 representam os arquivos de descrição de serviços gerados pelo *plugin* WTP, no Eclipse. Estes arquivos de descrição são passados para o *OWL-S Composer* através da ferramenta JAX-SA[Babik 2008] onde

os mesmos podem ser convertidos em arquivos de descrição OWL-S. Os arquivos OWL-S 1 e OWL-S 2 representam as descrições de serviços Web semânticos e são compostos graficamente (ref. Composição dos serviços) dentro do *OWL-S Composer*. No final, o *OWL-S composer* gera um OWL-S (OWL-S 3) referente a um arquivo de descrição de um serviço correspondente à composição criada.

2.1. OWL-S

A necessidade de descrições semânticas é essencial para a descoberta, invocação e composição automática de serviços Web. Estas características motivaram a utilização de uma linguagem de descrição que possibilitasse a uma máquina distinguir serviços com as mesmas características semânticas (objetivos similares) porém sintaticamente diferentes.

A Web Semântica favorece a resolução destes problemas, permitindo que dados e serviços sejam interpretados por software, sem o problema da ambigüidade [Berners-Lee et al. 2001]. Uma das linguagens que proporciona a descrição semântica de um serviço Web é a OWL-S (*Ontology Web Language for Service*) [Martin et al. 2007]. A OWL-S foi desenvolvida a partir da OWL (*Web Ontology Language*), que se trata de uma linguagem baseada em XML com a finalidade de descrever ontologias. Uma ontologia OWL consiste em um conjunto de classes, instâncias, propriedades e axiomas que descrevem um conceito dentro de um domínio particular [Murtagh 2007].

A OWL-S utiliza três ontologias para representar informações referentes a um serviço Web [Martin et al. 2007].

- *Profile*: Elemento que descreve as habilidades do serviço, facilitando sua descoberta automática. Através dessa ontologia são identificados elementos funcionais, precondições e efeitos, e elementos não funcionais como a categoria do serviço e informações sobre o provedor do serviço.
- *Model*: Esse elemento possui uma subclasse *Process* onde o comportamento do serviço é descrito. Basicamente, informações sobre invocações de serviços atômicos ou compostos durante a execução de um serviço e a troca de mensagens entre eles são descritas nesta ontologia. Um serviço composto pode ser definido como um serviço que contém informações de invocação de outros serviços atômicos ou compostos (ontologia *Model*), encapsulados em estruturas de controle como: *Sequence*, *SplitJoin*, *Anyorder*, etc. Já um serviço atômico, não pode invocar outros, somente pode ser executado.
- *Grounding*: Essa ontologia descreve o mapeamento das estruturas do *Model* em uma descrição WSDL, facilitando a execução automática dos serviços Web.

Assim, a linguagem OWL-S provê facilidades mediante a utilização do conceito da Web semântica para solucionar problemas relacionados à descoberta e à composição automática de serviços Web, sendo a linguagem utilizada neste trabalho pelo *OWL-S Composer*.

3. Requisitos de uma ferramenta para composição de serviços

Segundo [Chafle et al. 2007], alguns requisitos são essenciais para o desenvolvimento de uma ferramenta para compor serviços Web. Este trabalho avaliou alguns destes requisitos (*Funcionalidade*, *Interface*, *Usabilidade* e *Integração*), adaptando-os quando necessário, e propôs um quinto requisito: *Corretude*. A seguir são apresentados estes requisitos:

- Funcionalidade: O ambiente de desenvolvimento deve acompanhar o usuário durante todo o ciclo de vida de uma composição de serviços Web.
- Interface: Uma visão simplificada e funcional deve ser provida pelo ambiente de desenvolvimento de modo que o usuário não necessite utilizar nenhum outro recurso para realizar a composição de serviços.
- Usabilidade: O ambiente de desenvolvimento deve existir de forma que uma composição de serviços, realizada com sua utilização, possa ser intuitiva para um usuário inexperiente e prática para um usuário experiente, provendo uma capacidade de gerenciamento da composição de serviços Web da forma mais simples e eficiente possível.
- Integração: A integração entre os recursos do ambiente de desenvolvimento deve ser aproveitada utilizando a potencialidade de cada elemento nesse ambiente que possa contribuir para a composição de serviços Web.
- Corretude: O código gerado, referente à composição de serviços, em um ambiente de desenvolvimento de composições de serviços Web, deve ser legível e corresponder sintaticamente e semanticamente a um código válido correspondente gerado manualmente.

O desenvolvimento do *OWL-S Composer* atendeu a estes requisitos como é apresentado na próxima seção.

4. OWL-S Composer

Com o intuito de viabilizar a criação de uma ferramenta para gerenciar o desenvolvimento de composições de serviços Web semânticos foi desenvolvido um *plugin* para a plataforma Eclipse [Eclipse 2003] denominado *OWL-S Composer*.

O *OWL-S Composer* se baseia na utilização de um editor de diagramas que utiliza elementos em WSDL convertidos para a linguagem OWL-S ou serviços importados pelo usuário, de forma que a composição possa ser realizada visualmente para a posterior geração do código OWL-S referente ao serviço composto representado neste diagrama. A plataforma Eclipse foi utilizada devido a sua estrutura extensível, baseando-se fundamentalmente em componentes plugáveis. Todos os componentes do Eclipse, exceto o seu núcleo, são baseados em *plugins*. Assim, *OWL-S Composer* foi desenvolvido como um *plugin* podendo se integrar com outros *plugins* dessa plataforma.

O fato do OWL-S Composer ser um *plugin*, evidencia a conformidade deste com o requisito *integração*, proposto na seção 3, reforçando sua proposta de atuar em ambiente integrado para o desenvolvimento de Serviços Web. Com isso, nesse nesmo ambiente, serviços desenvolvidos utilizando o *plugin* WTP poderiam ser importados em uma composição através do uso da ferramenta externa *JAX-SA* [Babik 2008] que realiza a transformação dos documentos WSDL em descrições OWL-S correspondentes, ocorrendo, dessa forma, a integração do *OWL-S Composer* com o *plugin* WTP do Eclipse. Contudo, serviços produzidos pelo usuário manualmente ou através da utilização de outras ferramentas, também podem ser incluídos em uma composição, nesse caso, não necessitando da utilização do WTP ou do JAX-SA.

O *OWL-S Composer* (figura 2) se relaciona com os seguintes *plugins* da plataforma Eclipse: *Eclipse Modeling Framework Project* (EMF) [EMF 2008], *Graphical Editing Framework* (GEF) [GEF 2008], *Graphical Modeling Framework* (GMF)

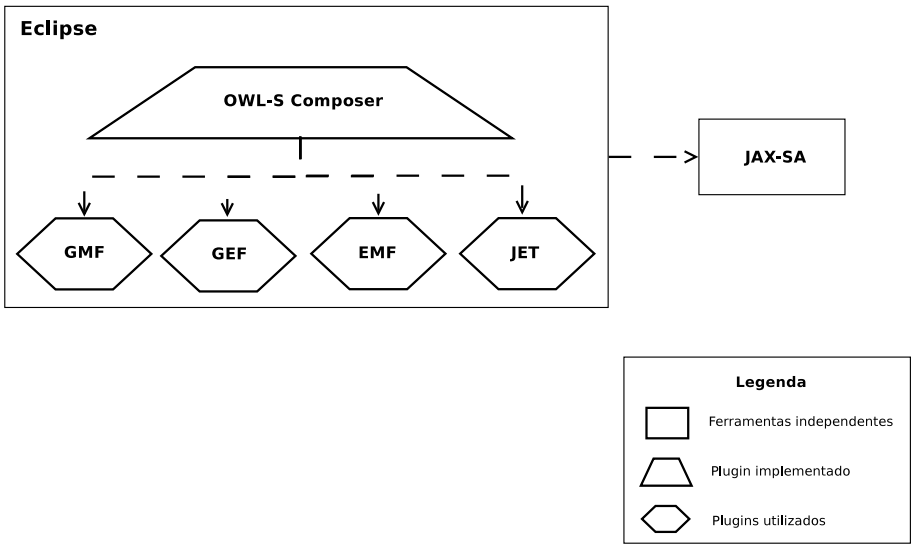

Figura 2. Arquitetura do *OWL-S Composer*

[GMF 2008], *Java Emitter Templates* (JET) [JET 2008]. Estes *plugins* são utilizados na criação e edição do diagrama e na geração do código referente à composição dos serviços, logo, os mesmos não foram implementados nesse trabalho apesar de servirem como base para a implementação do *OWL-S Composer*. Além disso, o *OWL-S Composer* se relaciona com a ferramenta JAX-SA, porém esta ferramenta não depende de nenhum elemento desta arquietura para sua execução, e, por isso, a ferramenta JAX-SA é representada como uma ferramenta independente na figura 2, juntamente com o Eclipse. A seguir são apresentadas mais algumas características do *OWL-S Composer* evidenciando os requisitos propostos na seção 3.

4.1. Interface gráfica

O *OWL-S Composer* deve garantir que o desenvolvimento de uma composição de serviços Web deve ser realizado de uma maneira intuitiva, em se tratando de um usuário inexperiente, e prática, em se tratando de um usuário experiente. Neste sentido, um *plugin* deve prover um alto nível de abstração sobre os detalhes técnicos referentes a essa atividade. Com o intuito de atender a este requisito, o *OWL-S Composer* utiliza uma abordagem ilustrada para compor serviços Web. A utilização desse tipo de abordagem é importante tanto em tempo de projeto quanto durante a manutenção de uma composição, auxiliando assim todo o seu ciclo de vida. Esta característica evidencia o requisito de *usabilidade* do *OWL-S Composer*. A figura 3 apresenta o editor visual do *OWL-S Composer*.

A figura 3 apresenta a utilização de uma janela contendo informações de projetos e outros tipos de arquivo suportados no Eclipse, denominada de *Package Explorer* (janela 1). O editor de diagramas é representado pela janela central (janela 2) contendo um espaço para a edição do diagrama e um componente chamado *Palette* onde pode-se selecionar os serviços OWL-S, as mensagens trocadas entre serviços, e os controladores dos serviços compostos tais como: *Sequence*, *Split* e *AnyOrder*. A janela intitulada *Properties* (janela 3) descreve as propriedades de cada elemento selecionado no diagrama, sendo que, algumas delas, podem ser editadas pelo usuário. A janela *Outline* (janela 4) é utilizada para dar uma visão geral do diagrama caso seu tamanho seja muito extenso. A posição dessas

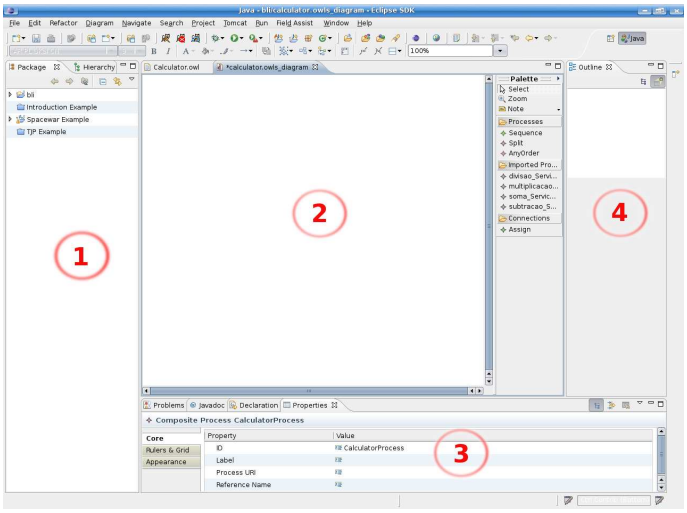

Figura 3. Projeto visual do *OWL-S Composer*

janelas pode ser modificada e outras janelas podem ser adicionadas dinamicamente de acordo com as necessidades do usuário.

Deve-se resaltar que o projeto da interface do *OWL-S Composer* visa ser completo em relação à composição de serviços Web, oferecendo recursos para que o usuário não necessite da utilização de outras telas para realização dessa atividade. Isso satisfaz o requisito de *interface* proposto na seção 3.

4.2. Composição de serviços Web

Na linguagem OWL-S, a modelagem de uma composição de serviços Web deve ser feita a partir de outros serviços atômicos ou compostos, dispostos em controladores de fluxo, com a finalidade de atingir um objetivo. No *OWL-S Composer* essa disposição dos serviços é modelada graficamente através do editor de diagramas (ver seção 4.1), sendo que os serviços disponíveis nesse diagrama devem ser selecionados pelo usuário no momento da criação do diagrama. Internamente, a biblioteca *OWL-S API* [MINDSWAP 2007] é utilizada no sentido de realizar o mapeamento das descrições semânticas dos serviços OWL-S em elementos do *OWL-S Composer*.

O diagrama de composição do *OWL-S Composer* pode desenvolver composições de serviços utilizando os serviços Web semânticos importados, devidamente indicados na janela de criação do diagrama, e os controladores de fluxo: *Sequence*, *Split* ou *AnyOrder*. A utilização de somente três controladores dentre os definidos na especificação da linguagem OWL-S 1.1 é somente ilustrativa, podendo ser estendida para os demais controladores em versões futuras deste *plugin*. Esta característica satisfaz parcialmente o requisito de *funcionalidade* pois o *OWL-S Composer*, em sua versão atual, não realiza o *deploy* automático das composições geradas. A figura 4 demonstra uma composição de Serviços Web utilizando o *OWL-S Composer*.

4.3. Geração de Código

Uma vez que a composição projetada estiver concluída no editor de diagramas, o usuário pode gerar o código referente a esta composição. O *plugin* JET é utilizado no *OWL-S*

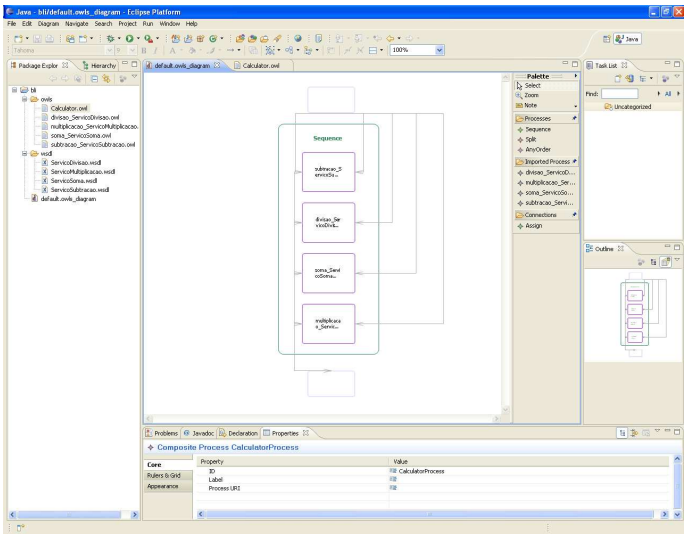

Figura 4. Demonstração de composição de serviços no *OWL-S Composer*

Composer para gerar o código correspondente a essa composição de serviços. No *OWL-S Composer*, o *template* do código da composição de serviços Web resultante é formatado seguindo a especificação da versão 1.1 da linguagem OWL-S. O requisito *corretude* pode ser evidenciado na validação sintática e semântica do código gerado pelo *OWL-S Composer*.

5. Trabalhos relacionados

Esta seção relaciona três ferramentas cujo propósito é compor serviços Web visualmente utilizando OWL-S: o *Malta's OWL-S Editor*, o *CMU's OWL-S Editor* e o *Protége OWL-S Editor*. Neste trabalho, as características de cada uma dessas ferramentas foram avaliadas e comparadas com o *OWL-S Composer*.

O *OWL-S Editor* [Scicluna et al. 2004] é uma ferramenta com o intuito de facilitar a criação de uma descrição de serviços Web utilizando OWL-S, sendo que, uma composição de serviços Web pode ser definida através de um diagrama de atividades baseado em UML. O *OWL-S Editor* fornece todos os recursos necessários da linguagem OWL-S para a criação de serviços atômicos, incluindo pré-condições e efeitos. Contudo, a completude dessa ferramenta não é verificada quando se trata da composição de serviços sendo que a mesma só possui suporte à duas estruturas de controle da linguagem OWL-S: o *Split* e o *IF-Then-Else*. Além disso por não ser um ambiente incorporado a uma IDE como Eclipse, este editor exije um conhecimento prévio na ferramenta e no ambiente de desenvolvimento.

O *CMU's OWL-S Development Environment* (CODE) [Srinivasan et al. 2006] é um *plugin* para a plataforma Eclipse com o objetivo de dar suporte ao desenvolvedor através de todo ciclo de vida do desenvolvimento de um serviço Web semântico [Srinivasan et al. 2006]. A representação gráfica das ontologias OWL-S nesse *plugin* é baseada em árvores de elementos, não propondo uma visualização gráfica completa da composição. A descrição OWL-S gerada através do WSDL2OWL-S é incompleta, necessitando de alguma edição antes de sua utilização, fator que dificulta a utilização deste *plugin*. Além disso, os arquivos WSDL são importados e não reutilizados do WTP, o que

evidencia uma integração incompleta com a plataforma de desenvolvimento Eclipse.

O *OWL-S Editor* [Knublauch et al. 2004] trabalha como um *plugin* para *Protégé OWL ontology editor* [Protégé 2006] que, por sua vez, provê um conjunto de ferramentas para o projeto de aplicações baseadas em conhecimento (*knowledge-based*) utilizando ontologias. Essa ferramenta, contudo, não habilita uma edição direta do diagrama de uma dada composição, sendo essa ação restrita a uma interface baseada em árvores de elementos. Além disso, a geração do código, referente à composição resultante, se mostrou desorganizada. Com isso, apesar dessa ferramenta fornecer melhores funcionalidades para manipulação de ontologias OWL-S em relação às ferramentas supra-citadas, a pouca interatividade na composição gráfica de serviços Web e as falhas na geração do código prejudicam sua usabilidade. Outro fator importante é que o Protègè não é uma IDE para desenvolvimento de serviços Web, o que dificulta a execução da composição de serviços previamente desenvolvidos.

Diante deste contexto, o *OWL-S Composer* se mostra como uma ferramenta mais adequada para o desenvolvimento de composições de serviços Web, enfatizando a utilização interativa de um ambiente gráfico, além da geração do código da composição de maneira clara e correta, como pode ser observado na seção 6.

A tabela 1 permite uma melhor visualização dos *plugins* analisados e uma comparação através dos requisitos propostos na seção 3.

Trab. Relacionados	**Funcionalidade**	**Interface**	**Usabilidade**	**Integração**	**Corretude**
OWL-S Composer	Parcial	Sim	Sim	Sim	Sim
Malta's OWL-S Ed.	Parcial	Sim	Sim	Não	Sim
CODE	Parcial	Sim	Não	Sim	Sim
Protégé OWL-S Ed.	Parcial	Sim	Sim	Sim	Parcial

Tabela 1. Quadro comparativo dos trabalhos relacionados.

6. Experimentos

Dois experimentos quanto ao código gerado referente à composição de serviços Web foram realizados com o intuito de validar o *OWL-S Composer*. Primeiramente foi realizado um experimento quanto à sintaxe do código gerado. O objetivo deste experimento foi verificar se o arquivo OWL-S gerado referente a uma composição correspondia a um arquivo OWL-S gerado manualmente. Neste sentido, critérios quanto a organização, estrutura das *tags* geradas e identação foram analizados. Um segundo experimento foi realizado quanto à semântica do código gerado ou seja, se esse era funcionalmente correto e se efetivamente executava uma composição de serviços Web. Em ambos os experimentos foram utilizados 4 serviços Web: *solicitaOrçamento*, *compraEquipamentoLoja*, *pagamentoCartao* e *entregaEquipamento*, construídos com o intuito de validar o *plugin*. A figura 5 apresenta esta composição de uma maneira gráfica utilizando o editor do *OWL-S Composer*.

A composição foi modelada de forma que o serviço *solicitaOrçamento* fosse inicialmente executado recebendo a quantidade e as características do computador a ser comprado. Os serviços *compraEquipamentoLoja*, *pagamentoCartao* e *entregaEquipamento* foram executados sequencialmente após o usuário ter selecionado um dos orçamentos.

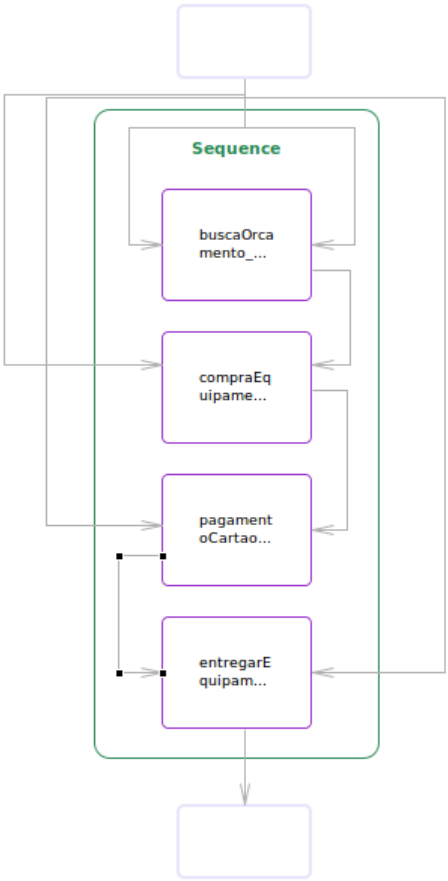

Figura 5. Compra de um equipamento utilizando o *OWL-S Composer*.

Após essa seleção, o serviço que realiza a compra na loja, *compraEquipamentoLoja*, recebe como entrada duas mensagens: o código do orçamento selecionado pelo usuário e o CPF do cliente que realiza a compra. Este serviço retorna um código da compra que é encaminhado para o serviço *pagamentoCartao*, onde juntamente com o número do cartão informado, o cliente efetua o pagamento. Em caso positivo da execução do serviço *pagamentoCartao*, o código do produto é repassado para o serviço de entrega, *entregaEquipamento*, assim como o endereço do cliente. Como retorno, este serviço fornece um código para que a entrega possa ser rastreada pelo cliente. As linhas presentes na figura 5 representam as entradas e saídas de cada serviço Web presente na composição.

6.1. Resultados

Os resultados obtidos foram analisados separadamente quanto à sintaxe e quanto à semântica. Sintaticamente, o código gerado através do *OWL-S Composer* correspondeu à organização e a estrutura das *tags* geradas manualmente.

Quanto à identação, a ontologia *Process* que corresponde aos serviços compostos apresentou uns pequenos desvios, não correspondendo de forma fidedigna ao código gerado manualmente. Este problema foi detectado como sendo pertinente à forma de utilização do *plugin* JET, porém, sua correção já foi incorporado como trabalho futuro deste *plugin*.

Quanto à semântica, foi realizado um *deploy* do código gerado automaticamente pelo *OWL-S Composer* para verificar se o mesmo executaria a composição que foi projetada graficamente. A execução da composição utilizou a OWL-S API na versão 1.1.0 e o resultado da composição da compra do equipamento foi obtido.

7. Conclusão

Nesse artigo foi apresentada uma ferramenta para gerenciar o desenvolvimento de uma composição de serviços utilizando o *OWL-S Composer*. O OWL-S Composer é um *plugin* desenvolvido para compor serviços Web semânticos através da plataforma Eclipse. Este *plugin* tem como principais características: a utilização de um ambiente visual para gerar as composições de serviços, a integração com outros *plugins* do Eclipse, como o WTP, minimizando a curva de aprendizagem e a facilidade de manipular as composições além de gerar o código da composição de uma maneira sintática e semanticamente correta. O OWL-S Composer foi analisado de acordo com os requisitos propostos neste trabalho e comparado com outros *plugins* e ferramentas que foram desenvolvidos com o intuito de compor serviços Web. O OWL-S Composer reuni os requisitos fundamentais: *funcionalidade*, *interface*, *usabilidade*, *integração* e *corretude*, o que facilita a sua incorporação e utilização em sistemas de informação em processo de desenvolvimento. Como trabalho futuro, pretende-se estender este *plugin* para a utilização de mecanismos de descoberta e composição automática de serviços Web utilizando, por exemplo, planejadores da Inteligência Artificial.

Referências

Babik, M. (2008). Jax-sa. Último acesso em 9 de junho de 2008.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific American*.

Chafle, G., Das, G., Dasgupta, K., Kumar, A., Mittal, S., Mukherjea, S., and Srivastava, B. (2007). An integrated development enviroment for web service composition. In *IEEE ICWS 2007*, pages 1–9.

Claro, D. B. and Macêdo, R. J. A. (2008a). Dependable web service compositions using a semantic replication scheme. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuidos (SBRC'2008)*, Rio de Janeiro,RJ.

Claro, D. B. and Macêdo, R. J. A. (2008b). Web services e sua relação com sistemas de informação. In *Anais do Simpósio Brasileiro de Sistemas de Informação(SBSI'08)*, pages 356–359, Rio de Janeiro, RJ.

Dai, N., Mandel, L., and Ryman, A. (2007). *Eclipse Web Tools Platform: Developing Java(TM) Web Applications*. Addison-Wesley Professional, 1 edition.

Digiampietri, L. A., Pérez-Alcázar, J. J., and Bauzer, C. M. (2007). Ai planning in web services composition: a review of current approaches and a new solution. In *VI Encontro de Inteligência Artificial. Anais do XXVII Congresso da SBC 2007*.

Domingue, J., Lausen, H., and Fensel, D. (2005). Web service modeling ontology.

Eclipse (2003). Eclipse platform technical overview. Object Technology International. www.eclipse.org/whitepapers/eclipse-overview.pdf.

EMF (2008). Eclipse modeling framework project. Último acesso em 24 de maio de 2008.

GEF (2008). Graphical editing framework. Último acesso em 24 de maio de 2008.

GMF (2008). Graphical modeling framework. Último acesso em 24 de maio de 2008.

JET (2008). Java emitter templates. Último acesso em 24 de maio de 2008.

Knublauch, H., Fergerson, R., Noy, N., and Musen, M. (2004). The protégé owl plugin: An open developoment environment for semantic web applications. In *Proceedings of the 3rd International Semantic Web Conference (ISWC 2004)*, pages 229–243. Springer/LNCS 3298.

Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McilRaith, S., NaraYanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., and Sycara, K. (2007). Owl-s: Semantic markup for web services.

MINDSWAP (2007). Maryland information and network dynamics lab semantic web agents project.

Murtagh, D. (2007). *Automated Web Services Composition*. PhD thesis, University of Dublin, Dublin.

Protégé (2006). Protégé owl editor. Último acesso em 29 de maio de 2008.

Scicluna, J., Abela, C., and Montebello, M. (2004). Visual modelling of owl-s services. In *Proceedings of the the IADIS International Conference WWW/Internet*.

Srinivasan, N., Paolucci, M., and Sycara, K. (2006). Semantic web service discovery in the owl-s ide. In *Proceedings of the IEEE 39th Hawaii International Conference on System Sciences*.